REMARKS

Status:

Claims 1-11 stand rejected under 35 U.S.C. §101 as being directed to nonstatutory subject matter. Claims 1, 4, 6-9, 11-12, 15, 18-21, 23-27, 30 and 37 stand rejected under 35 U.S.C. §102(b) as being anticipated by the teaching of the Aho reference. Claims 1, 6-9, 12, 24, 27, 32-35 stand rejected under 35 U.S.C. §102(e) as being anticipated by the teaching of US Pat. No. 6,378,126 (hereinafter Tang). Claims2-3, 5, 13-14, 16-17, 28-29 and 31 stand rejected under 35 U.S.C. §103(a) as being unpatentable over the teaching of the Aho reference considered in view of US Pat. No. 5,325,531 (hereinafter McKeeman).

Claims 1-37 are presented for reconsideration as is explained in the discussion that follows.

Analysis:

Claims 1-11 have been amended to clarify that the method is executed on a computer system and, as so clarified, these claims are believed to be statutory. Accordingly, it is respectfully requested that the rejection under 35 U.S.C. §101 be withdrawn.

Applicant has developed a specific, computer implemented approach to testing code for valid syntax. It utilizes a syntax tree form with a single start or root node and a single end node.
Tokens are initially identified and recorded, preferably in a table. The approach advances through the code using a particular methodology, which moves a designated current node, starting with the root node and advancing through the tree to the single

end node. Each advance involves returning potential current nodes and comparing those nodes for identified tokens, which when found permit advancing to that node as the new current node. The process repeats until the end node is reached. This process is, for example, specified in claim 1. Refinements, such as tables, to facilitate the process as regards tracking tokens found as the process progresses are afforded protection in claims 2 and 3.

Claim 4 emphasizes the number of nodes occurring between a node under consideration and the current node as indicating distance. Such approachs to validating code syntax are similarly reflected in the claim sets of the other independent claims 12, 24 and 27.

Now looking to the Office Action at p. 4, a side by side presentation of Applicants claim 1 and Aho compares the syntax tree of claim 1 to the parse tree taught by Aho at Aho p. 29. But Aho's parse tree is like the prior art tree of Applicant's Fig. 1 not the single end node diagram of Fig. 2 that is described in claim 1, which calls for "*an end node, such that all paths through the tree lead to the end node.*" (Italics added to indicate a quote). The office Action at p. 4 also cites p. 98 of Aho for recognition of tokens, but again the diagram (see Aho Fig. 3.12) fans out rather than define an end node. At Office action p. 5 reference is made to Aho p. 100 for the process but the multi branch process and method of failure recognition are not those called for in Applicant's claim. It is also noted that Aho at p. 114 in listing five aspects of the model calls for one start point at 4, but plural final states at 5. This is not Applicant's claimed model.

At Office Action p. 5, regarding Applicant's claim 4, it is not clear what treating as a distance the minimum number of insertions and deletions to convert one text string into another (striped to tiger) has to do with counting intermediate nodes between a current and prospective node as a measure of distance.

Now looking to the side-by- side at page 12 of the Office Action, col. 1 lines 31-37 of the Tang teaching is discussed. But, that is a prior art discussion and only says that the semantics of the tokens are checked for errors. There is no description as to how they are checked. Then col. 5 lines 37-40 is discussed, which describes a preferred parse tree layout. But , again, there is not description of a process for checking syntax.

Indeed, it does not appear that Tang is directed to or incidentally describes a process for syntax validation. Tang relates to code that is written in two languages. Hence the last paragraph of the of Tang's "Background of the Invention" indicates: "*There is a need in the art for an improved method and system for processing and compiling SQL statements embedded in a host language program.*"

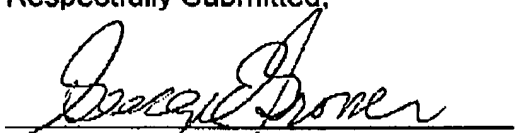
At the sixth paragraph of Tang's "Summary of the Preferred Embodiments", it is indicated that: "*Preferred embodiments provide an improved language processor because the SQL statements are converted to API function calls and then parsed in the same manner that statements in the first language are parsed in the parse tree before being converted into target code.* ***In this way, the embedded language statements are subjected to the same optimization techniques applied to the host language***" (bolding added for emphasis). Hence, it appears the syntax validation is provided by the host language compiler or , possibly at box 26 of Fig. 2 which is not described in any detail. There is parse tree creation discussed but where is there any description of the syntax validation process?

The body of Applicant's independent claims 1, 12, 24 and 27 describes an iterative approach to validate syntax. Where in the Aho or Tang teaching is there a description or suggestion of the steps specified in claim 1 to assess validity through an iterative advance to an end node? None of the other references make up for this deficiency. McKeeman teaches token tables but there is no column indicated to support syntax

validation let alone a discussion of the steps described and claimed by Applicant for such validation.

In accordance with the foregoing, it is believed that Applicant has described and claimed an inventive contribution to syntax validation. Applicant therefore respectfully requests that the rejection of claims be withdrawn and early notice be given that this case is in condition for allowance.

Respectfully Submitted,



George E. Grosser
Reg. No. 25,629l

c/o IBM Corp.
Dept. T81/Bldg. 503 PO Box 12195
Research Triangle Park, NC 27709

Tel. No.(919)968-7847 Fax 919-254-4330
EMAIL: gegch@prodigy.net